MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2015/2016

ECP2036 – MICROPROCESSOR SYSTEMS AND INTERFACING

(ME)

17 OCTOBER 2015
2:30 P.M. – 4:30 P.M.
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This Question paper consists of 7 pages with 4 questions only.
2. Attempt **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please write all your answers in the Answer Booklet provided.
4. Opcode map and Special Function Register formats are provided in Appendices.

Question 1

a) An 8051 microcontroller has to access 32kBytes of external memory. Determine the number of address and data lines to be used. [3 marks]

b) Describe the function of these 8051 control pins.

(i) $\overline{PSEN}$ [2 marks]
(ii) ALE [2 marks]
(iii) $\overline{EA}$ [2 marks]
(iv) RST [2 marks]

c) An 8051 microcontroller-based system is to be designed requiring 32kBytes of RAM. 16kBytes of RAM memory blocks are available.

(i) Evaluate the number of RAM memory block required. [1 mark]
(ii) Determine the address range of each memory block used. [4 marks]
(iii) Draw the configuration of the system showing the 8051signal lines to be used for the address, data and control buses. [9 marks]

Question 2

a) The following is an 8051 microcontroller's instruction.

MOV 43H, #0A2H

(i) State the addressing mode of this instruction. [1 mark]
(ii) Explain the purpose of each byte of this instruction.. [2 marks]
(iii) If an 8051 is operating from 16MHz crystal, how long does this instruction takes to execute? [2 marks]

b) Complete the following list file (.lst) by filling in the missing data.

No.	Address	Machine Code	Instruction
1	0A00		ORG 0A00H
2	0A00		ADD A,R5
3			ORL A,#2BH
4			MOVC A, @A+DPTR
5			SETB 30H
6			DEC R5

[10 marks]

c) Internal memory locations from 40H to 49H contain the numbers 0 to 9 respectively. By using PUSH and POP instructions, write the assembly language instructions to reverse the order in which the number are stored (0 is put in 49H, 1 in 48H, etc.) [10 marks]

Continued...

Question 3

a) A string of 7-bit ASCII code is stored in external memory of 8051 (starting address of the string is 4000H). The string is terminated by a byte contained 00H.

Write an assembly language instruction sequence to transfer the string to a personal computer via serial port. The serial port should be initialized in 8-bit UART with an added (even) parity bit as bit 7. The baud rate (9600) should be generated by the Timer 1. Assume 11.059MHz operating frequency is used. [15 marks]

b) Name the special function registers to control the 8051 interrupts and the interrupt priorities. What should be the setting values of the special function registers if Timer 0 and Counter 1 interrupts are both enabled with Counter 1 has higher priority? [6 marks]

c) In an 8051 door logging system, Timer 0 is used to emulate the Real-Time-Clock operation and External Interrupt 1 is used to detect the door opening through an IR sensor. Which interrupt service should be given top priority in order avoid loss of accuracy? Justify your answer. [4 marks]

Question 4

The concrete mixing system shown in *Figure 1* is to be controlled by an 8051 microcontroller, which involves performing the following process:

- The tank is first filled with water through a solenoid Valve W.
- When the water reaches Level W, Valve W is closed and the tank is now filled with cement through Valve C.
- When the mixture in the tank reaches Level C, Valve C is closed and the tank is then filled with sand through Valve S.
- When the mixture in the tank reaches Level S, Valve S is closed
- The mixer motor starts for approximately 3 minutes.
- After that, the drainage Valve E opens to empty the tank.
- When the mixture reaches Level E, Valve E is closed and the whole process is repeated after 1 minute.

The tank has four level sensors that send signals to input lines P1.0 to P1.3. A logical low from the sensor indicates that the level has been reached. The output lines P0.0 to P0.3 provide signals to the solenoid valves. A logical low from the lines will open the corresponding valve. The output lines P0.4 provide signals to the mixer motor which is also activated by a logical low. Write an assembly language instruction sequence to carry out the process. [25 marks]

Continued...

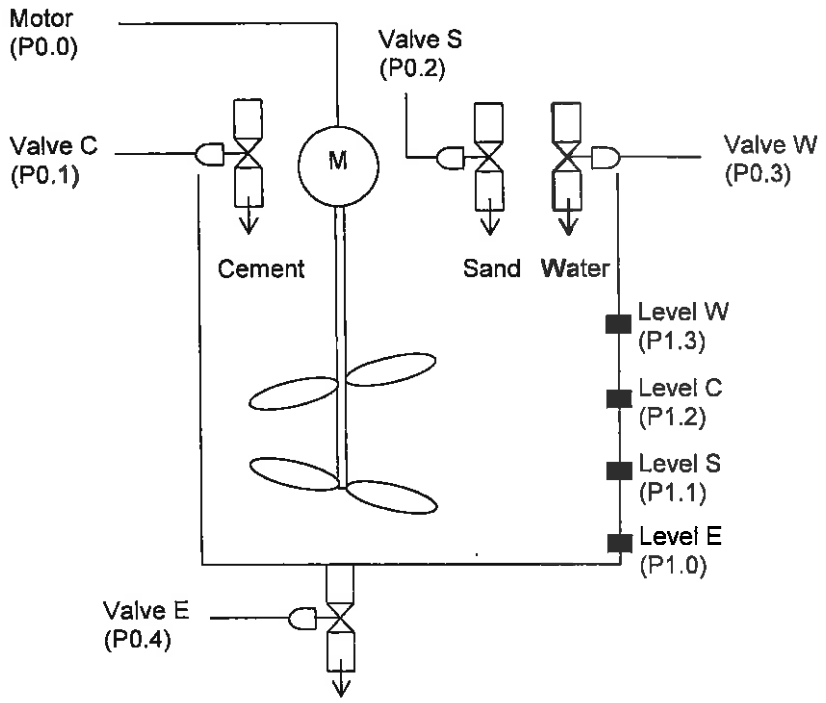
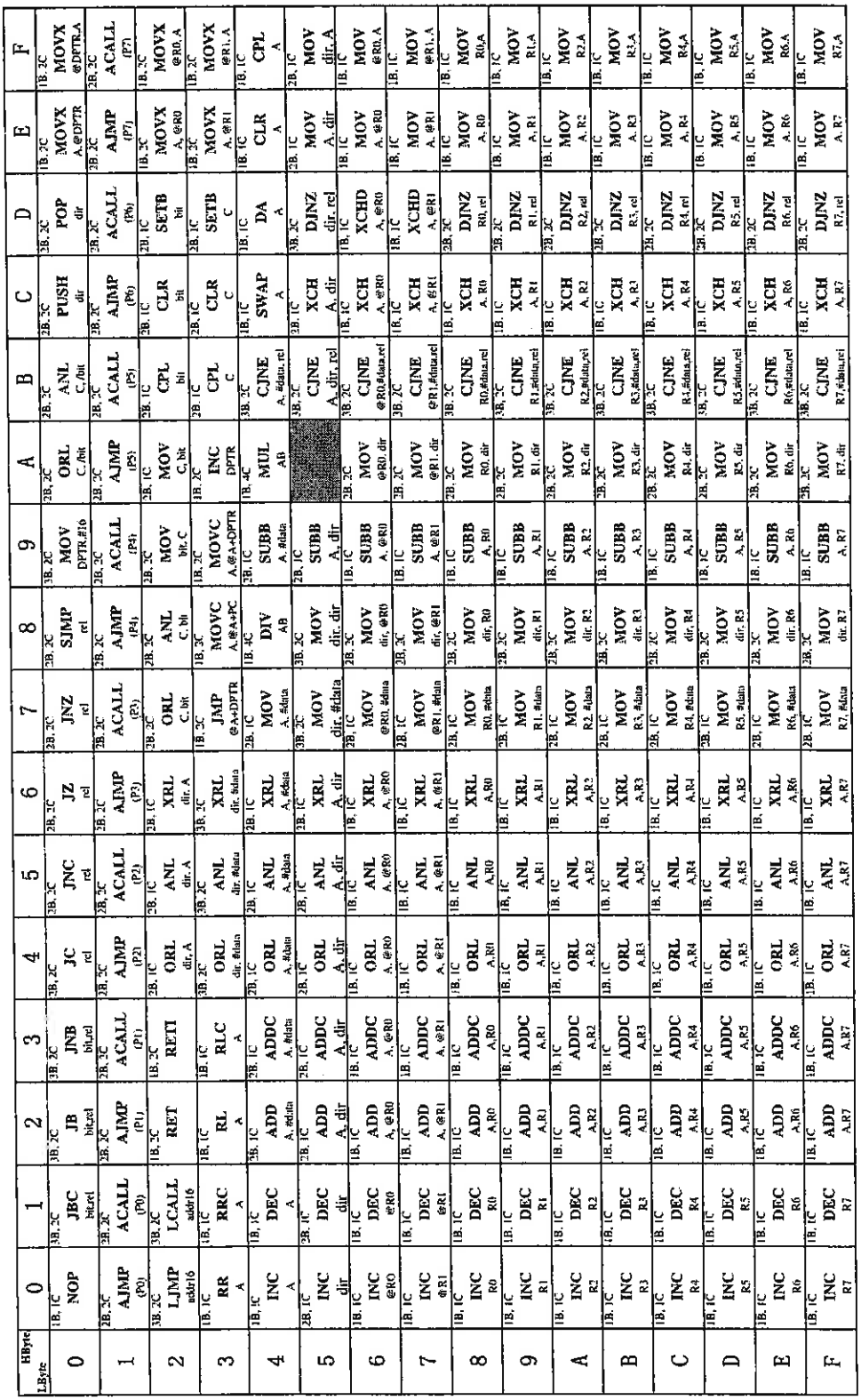


Figure 1

Continued...

Appendix A: Opcode Map



HByte / LByte	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1B, 1C **NOP**	3B, 2C **JBC** bit,rel	3B, 2C **JB** bit,rel	3B, 2C **JNB** bit,rel	3B, 2C **JC** rel	3B, 2C **JNC** rel	2B, 2C **JZ** rel	2B, 2C **JNZ** rel	2B, 2C **SJMP** rel	3B, 2C **MOV** DPTR,#16	2B, 2C **ORL** C,/bit	2B, 2C **ANL** C,/bit	2B, 2C **PUSH** dir	3B, 2C **POP** dir	1B, 2C **MOVX** A,@DPTR	1B, 2C **MOVX** @DPTR,A
1	2B, 2C **AJMP** (P0)	2B, 2C **ACALL** (P0)	2B, 2C **AJMP** (P1)	2B, 2C **ACALL** (P1)	2B, 2C **AJMP** (P2)	2B, 2C **ACALL** (P2)	2B, 2C **AJMP** (P3)	2B, 2C **ACALL** (P3)	2B, 2C **AJMP** (P4)	2B, 2C **ACALL** (P4)	2B, 2C **AJMP** (P5)	2B, 2C **ACALL** (P5)	2B, 2C **AJMP** (P6)	2B, 2C **ACALL** (P6)	2B, 2C **AJMP** (P7)	2B, 2C **ACALL** (P7)
2	3B, 2C **LJMP** addr16	3B, 2C **LCALL** addr16	1B, 2C **RET**	1B, 2C **RETI**	2B, 1C **ORL** dir, A	2B, 1C **ANL** dir, A	2B, 1C **XRL** dir, A	2B, 2C **ORL** C, bit	2B, 2C **ANL** C, bit	2B, 2C **MOV** bit, C	2B, 1C **MOV** C, bit	2B, 1C **CPL** bit	2B, 1C **CLR** bit	2B, 1C **SETB** bit	1B, 2C **MOVX** A, @R0	1B, 2C **MOVX** @R0, A
3	1B, 1C **RR** A	1B, 1C **RRC** A	1B, 1C **RL** A	1B, 1C **RLC** A	3B, 2C **ORL** dir, #data	3B, 2C **ANL** dir, #data	3B, 2C **XRL** dir, #data	1B, 2C **JMP** @A+DPTR	1B, 3C **MOVC** A,@A+PC	1B, 2C **MOVC** A,@A+DPTR	1B, 2C **INC** DPTR	2B, 1C **CPL** C	2B, 1C **CLR** C	2B, 1C **SETB** C	1B, 2C **MOVX** A, @R1	1B, 2C **MOVX** @R1, A
4	1B, 1C **INC** A	1B, 1C **DEC** A	2B, 1C **ADD** A, #data	2B, 1C **ADDC** A, #data	2B, 1C **ORL** A, #data	2B, 1C **ANL** A, #data	2B, 1C **XRL** A, #data	2B, 1C **MOV** A, #data	1B, 4C **DIV** AB	2B, 1C **SUBB** A, #data	1B, 4C **MUL** AB	3B, 2C **CJNE** A, #data, rel	1B, 1C **SWAP** A	1B, 1C **DA** A	1B, 1C **CLR** A	1B, 1C **CPL** A
5	2B, 1C **INC** dir	2B, 1C **DEC** dir	2B, 1C **ADD** A, dir	2B, 1C **ADDC** A, dir	2B, 1C **ORL** A, dir	2B, 1C **ANL** A, dir	2B, 1C **XRL** A, dir	3B, 2C **MOV** dir, #data	3B, 2C **MOV** dir, dir	2B, 1C **SUBB** A, dir	[redacted]	3B, 2C **CJNE** A, dir, rel	2B, 1C **XCH** A, dir	3B, 2C **DJNZ** dir, rel	2B, 1C **MOV** A, dir	2B, 1C **MOV** dir, A
6	1B, 1C **INC** @R0	1B, 1C **DEC** @R0	1B, 1C **ADD** A, @R0	1B, 1C **ADDC** A, @R0	1B, 1C **ORL** A, @R0	1B, 1C **ANL** A, @R0	1B, 1C **XRL** A, @R0	2B, 1C **MOV** @R0, #data	2B, 3C **MOV** dir, @R0	1B, 1C **SUBB** A, @R0	2B, 2C **MOV** @R0, dir	3B, 2C **CJNE** @R0,#data,rel	1B, 1C **XCH** A, @R0	1B, 1C **XCHD** A, @R0	1B, 1C **MOV** A, @R0	1B, 1C **MOV** @R0, A
7	1B, 1C **INC** @R1	1B, 1C **DEC** @R1	1B, 1C **ADD** A, @R1	1B, 1C **ADDC** A, @R1	1B, 1C **ORL** A, @R1	1B, 1C **ANL** A, @R1	1B, 1C **XRL** A, @R1	2B, 1C **MOV** @R1, #data	2B, 3C **MOV** dir, @R1	1B, 1C **SUBB** A, @R1	2B, 2C **MOV** @R1, dir	3B, 2C **CJNE** @R1,#data,rel	1B, 1C **XCH** A, @R1	1B, 1C **XCHD** A, @R1	1B, 1C **MOV** A, @R1	1B, 1C **MOV** @R1, A
8	1B, 1C **INC** R0	1B, 1C **DEC** R0	1B, 1C **ADD** A,R0	1B, 1C **ADDC** A,R0	1B, 1C **ORL** A,R0	1B, 1C **ANL** A,R0	1B, 1C **XRL** A,R0	2B, 1C **MOV** R0, #data	2B, 2C **MOV** dir, R0	1B, 1C **SUBB** A, R0	2B, 2C **MOV** R0, dir	3B, 2C **CJNE** R0,#data,rel	1B, 1C **XCH** A, R0	2B, 2C **DJNZ** R0, rel	1B, 1C **MOV** A, R0	1B, 1C **MOV** R0,A
9	1B, 1C **INC** R1	1B, 1C **DEC** R1	1B, 1C **ADD** A,R1	1B, 1C **ADDC** A,R1	1B, 1C **ORL** A,R1	1B, 1C **ANL** A,R1	1B, 1C **XRL** A,R1	2B, 1C **MOV** R1, #data	2B, 3C **MOV** dir,R1	1B, 1C **SUBB** A, R1	2B, 2C **MOV** R1, dir	3B, 2C **CJNE** R1,#data,rel	1B, 1C **XCH** A, R1	2B, 2C **DJNZ** R1, rel	1B, 1C **MOV** A, R1	1B, 1C **MOV** R1,A
A	1B, 1C **INC** R2	1B, 1C **DEC** R2	1B, 1C **ADD** A,R2	1B, 1C **ADDC** A,R2	1B, 1C **ORL** A,R2	1B, 1C **ANL** A,R2	1B, 1C **XRL** A,R2	2B, 1C **MOV** R2, #data	2B, 3C **MOV** dir, R2	1B, 1C **SUBB** A, R2	2B, 2C **MOV** R2, dir	3B, 2C **CJNE** R2,#data,rel	1B, 1C **XCH** A, R2	2B, 2C **DJNZ** R2, rel	1B, 1C **MOV** A, R2	1B, 1C **MOV** R2,A
B	1B, 1C **INC** R3	1B, 1C **DEC** R3	1B, 1C **ADD** A,R3	1B, 1C **ADDC** A,R3	1B, 1C **ORL** A,R3	1B, 1C **ANL** A,R3	1B, 1C **XRL** A,R3	2B, 1C **MOV** R3, #data	2B, 3C **MOV** dir, R3	1B, 1C **SUBB** A, R3	2B, 2C **MOV** R3, dir	3B, 2C **CJNE** R3,#data,rel	1B, 1C **XCH** A, R3	2B, 2C **DJNZ** R3, rel	1B, 1C **MOV** A, R3	1B, 1C **MOV** R3,A
C	1B, 1C **INC** R4	1B, 1C **DEC** R4	1B, 1C **ADD** A,R4	1B, 1C **ADDC** A,R4	1B, 1C **ORL** A,R4	1B, 1C **ANL** A,R4	1B, 1C **XRL** A,R4	2B, 1C **MOV** R4, #data	2B, 3C **MOV** dir, R4	1B, 1C **SUBB** A, R4	2B, 2C **MOV** R4, dir	3B, 2C **CJNE** R4,#data,rel	1B, 1C **XCH** A, R4	2B, 2C **DJNZ** R4, rel	1B, 1C **MOV** A, R4	1B, 1C **MOV** R4,A
D	1B, 1C **INC** R5	1B, 1C **DEC** R5	1B, 1C **ADD** A,R5	1B, 1C **ADDC** A,R5	1B, 1C **ORL** A,R5	1B, 1C **ANL** A,R5	1B, 1C **XRL** A,R5	2B, 1C **MOV** R5, #data	2B, 3C **MOV** dir, R5	1B, 1C **SUBB** A, R5	2B, 2C **MOV** R5, dir	3B, 2C **CJNE** R5,#data,rel	1B, 1C **XCH** A, R5	2B, 2C **DJNZ** R5, rel	1B, 1C **MOV** A, R5	1B, 1C **MOV** R5,A
E	1B, 1C **INC** R6	1B, 1C **DEC** R6	1B, 1C **ADD** A,R6	1B, 1C **ADDC** A,R6	1B, 1C **ORL** A,R6	1B, 1C **ANL** A,R6	1B, 1C **XRL** A,R6	2B, 1C **MOV** R6, #data	2B, 3C **MOV** dir, R6	1B, 1C **SUBB** A, R6	2B, 2C **MOV** R6, dir	3B, 2C **CJNE** R6,#data,rel	1B, 1C **XCH** A, R6	2B, 2C **DJNZ** R6, rel	1B, 1C **MOV** A, R6	1B, 1C **MOV** R6,A
F	1B, 1C **INC** R7	1B, 1C **DEC** R7	1B, 1C **ADD** A,R7	1B, 1C **ADDC** A,R7	1B, 1C **ORL** A,R7	1B, 1C **ANL** A,R7	1B, 1C **XRL** A,R7	2B, 1C **MOV** R7, #data	2B, 3C **MOV** dir, R7	1B, 1C **SUBB** A, R7	2B, 2C **MOV** R7, dir	3B, 2C **CJNE** R7,#data,rel	1B, 1C **XCH** A, R7	2B, 2C **DJNZ** R7, rel	1B, 1C **MOV** A, R7	1B, 1C **MOV** R7,A

Continued...

Appendix B: Special Function Register Format

TMOD : [Bit 0 (LSB) to Bit 3 is for Timer 0 and Bit 4 to Bit 7 (MSB) is for Timer 1]

GATE	C / /T	M1	M0	GATE	C / /T	M0	M1

GATE: Timer only runs while /INT1 is set.
CI IT: '1' for event counter, '0' for interval timer
M1, MO: Mode bit select
"00" Mode 0 – 13-bit timer mode
"01" Mode 1 – 16-bit timer mode
"10" Mode 2 – 8-bit auto-reload mode
"11" Mode 3 – Split timer mode

TCON :

TF1	TR1	TFO	TRO	IE1	IT1	IE0	IT0

TCON.7 TF1 Timer 1 overflow flag. Set by hardware on overflow. Clear by hardware when processor vectors to interrupt routine.
TCON.6 TRI Timer 1 run control bit. Set/cleared by software to start/stop timer.
TCON.5 TFO Timer 0 overflow flag. Set by hardware on overflow. Clear by hardware when processor vectors to interrupt routine.
TCON.4 TRO Timer 0 run control bit. Set/cleared by software to start/stop timer.
TCON.3 IE1 Interrupt 1 Edge flag. Set by hardware when interrupt 1 falling edge is detected. Cleared when interrupt is processed.
TCON.2 IT1 Interrupt 1 Type control bit. Set / cleared by software to specify falling edge / low level triggered external interrupts.
TCON. 1 IEO Interrupt 0 Edge flag. Set by hardware when interrupt 1 falling edge is detected. Cleared when interrupt is processed.
TCON.0 IT0 Interrupt 0 Type control bit. Set / cleared by software to specify falling edge / low level triggered external interrupts.

SCON :

SMO	SM1	SM2	REN	TB8	RB8	TI	RI

SMO	SM1	
0	0	= Shift register mode
0	1	= 8-bit UART mode
1	0	= 9-bit UART mode (Fixed Baud Rate)
1	1	= 9-bit UART mode (Variable Baud Rate)

SM2 = '1' = Enable multiprocessor communication
REN = Receiver Enable
TB8 = Transmit Bit
TI = Transmit Interrupt
RI = Receive Interrupt

Continued...

IE:

EA		ET2	ES	ET1	EX1	ET0	EXO

Bit Position	Symbol	Bit Address	Description
IE.7	EA	AFH	Global enable/disable. EA ='1', each individual source is enable/disable By seetting/clearing its enable bit. EA = 'O', disable all interrupts.
IE.6	-	AEH	Undefined
IE.5	-	ADH	Not implemented in 805 1. ET2 for 8052.
IE.4	ES	ACH	Serial port interrupt enable bit.
IE.3	ET1	ABH	Timer1 interrupt enable bit.
IE.2	EX1	AAH	External interrupt enable bit.
IE. 1	ET0	A9H	Timer0 interrupt enable bit.
IE.0	EXO	A8H	External interrupt enable bit.

IP:

		PT2	PS	PT1	PX1	PT0	PX0

IP.7	-	-	Undefined.
IP.6	-	-	Undefined.
IP.5	-	BDH	Not implemented in 8051. PT2 for 8052.
IP.4	PS	BCH	Serial port interrupt priority bit.
IP.3	PT1	BBH	Timer1 interrupt priority bit.
IP.2	PX1	BAH	External interrupt priority bit.
IP.1	PT0	B9H	Timer-0 interrupt priority bit.
IP.0	PX0	B8H	External interrupt priority bit.

Selected Interrupt Vectors

Interrupt source	Flag	Vector Address
System Reset	RST	0000H
External 0	IE0	0003H
Timer 2 (8052)	TF2 & EXF2	002BH

PSW:

CY	AC	F0	RS1	RS0	OV	-	P

CY : Carry Flag
AC : Auxiliary Carry Flag
RS 1, RS0: Register Bank Select
OV : Overflow Flag
P : Parity

End of Paper